statistics

February 1, 2021

[62]:

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from matplotlib.gridspec import GridSpec
%matplotlib inline

import datetime
import geopandas as gpd

import numpy as np
import seaborn as sns
```

[63]:

```
import warnings
warnings.filterwarnings("ignore")

pd.set_option("display.float_format",lambda x: "%.2f" % x)
```

[64]:

```
#importing data using pandas

df_covid = pd.read_csv("covidde/covid_de.csv")
df_dmo = pd.read_csv("covidde/demographics_de.csv")


#reading .shp (shapeFile) file using geopandas. geopandas extends the datatypes␣
 ↪used by pandas
df_mp = gpd.read_file("covidde/de_state.shp")
df_mp_country = gpd.read_file("covidde/de_county.shp")
```

[65]:

```
df_covid
```

[65]:

```
                     state             county age_group gender        date  \
0        Baden-Wuerttemberg  LK Alb-Donau-Kreis     00-04      F  2020-03-27
1        Baden-Wuerttemberg  LK Alb-Donau-Kreis     00-04      F  2020-03-28
2        Baden-Wuerttemberg  LK Alb-Donau-Kreis     00-04      F  2020-04-03
3        Baden-Wuerttemberg  LK Alb-Donau-Kreis     00-04      F  2020-10-18
4        Baden-Wuerttemberg  LK Alb-Donau-Kreis     00-04      F  2020-10-22
…                        …                  …        …    …          …
```

```
416309          Thueringen           SK Weimar     80-99      M  2020-12-26
416310          Thueringen           SK Weimar     80-99      M  2020-12-28
416311          Thueringen           SK Weimar     80-99      M  2020-12-30
416312          Thueringen           SK Weimar     80-99      M  2020-12-31
416313          Thueringen           SK Weimar       NaN      F  2020-12-31

        cases  deaths  recovered
0           1       0          1
1           1       0          1
2           1       0          1
3           1       0          1
4           1       0          1
…       …     …        …
416309      4       1          0
416310      1       0          0
416311      2       0          0
416312      1       0          0
416313      1       0          0

[416314 rows x 8 columns]
```

[66]:

```
#changing the name of state column value

df_covid["state"] = df_covid["state"].replace("Baden-Wuerttemberg",␣
 ↪"Baden-Württemberg")
df_covid["state"] = df_covid["state"].replace("Thueringen", "Thüringen")
```

[67]:

```
df_covid
```

[67]:

```
                    state              county age_group gender        date  \
0       Baden-Württemberg  LK Alb-Donau-Kreis     00-04      F  2020-03-27
1       Baden-Württemberg  LK Alb-Donau-Kreis     00-04      F  2020-03-28
2       Baden-Württemberg  LK Alb-Donau-Kreis     00-04      F  2020-04-03
3       Baden-Württemberg  LK Alb-Donau-Kreis     00-04      F  2020-10-18
4       Baden-Württemberg  LK Alb-Donau-Kreis     00-04      F  2020-10-22
…                   …                 …       …    …        …
416309          Thüringen           SK Weimar     80-99      M  2020-12-26
416310          Thüringen           SK Weimar     80-99      M  2020-12-28
416311          Thüringen           SK Weimar     80-99      M  2020-12-30
416312          Thüringen           SK Weimar     80-99      M  2020-12-31
416313          Thüringen           SK Weimar       NaN      F  2020-12-31

        cases  deaths  recovered
0           1       0          1
1           1       0          1
2           1       0          1
3           1       0          1
```

```
4           1       0          1
…       …     …        …
416309      4       1          0
416310      1       0          0
416311      2       0          0
416312      1       0          0
416313      1       0          0

[416314 rows x 8 columns]
```

[68]:

```
#formatting date value
df_covid["date"] = pd.to_datetime(df_covid["date"])
```

[69]:

```
#changing the name of state column value

df_dmo["state"] = df_dmo["state"].replace("Baden-Wuerttemberg",␣
 ↪"Baden-Württemberg")
df_dmo["state"] = df_dmo["state"].replace("Thueringen", "Thüringen")
```

[70]:

```
#changing the value of gender column of df_dmp dataframe

df_dmo["gender"] = np.where(df_dmo["gender"] == "female", "F", "M" )
```

[71]:

```
df_dmo
```

[71]:

```
                 state gender age_group  population
0    Baden-Württemberg      F     00-04      261674
1    Baden-Württemberg      F     05-14      490822
2    Baden-Württemberg      F     15-34     1293488
3    Baden-Württemberg      F     35-59     1919649
4    Baden-Württemberg      F     60-79     1182736
..                 …    …        …          …
187          Thüringen      M     05-14       92545
188          Thüringen      M     15-34      214553
189          Thüringen      M     35-59      384822
190          Thüringen      M     60-79      264189
191          Thüringen      M     80-99       57340

[192 rows x 4 columns]
```

[72]:

```
#finding 'NA' values and displaying them"

df_covid[(df_covid["gender"].isnull()) | (df_covid["age_group"].isnull())]
```

[72]:

```
                     state              county age_group gender       date  \
200      Baden-Württemberg  LK Alb-Donau-Kreis     05-14    NaN 2020-10-30
201      Baden-Württemberg  LK Alb-Donau-Kreis     05-14    NaN 2020-11-19
```

```
517     Baden-Württemberg  LK Alb-Donau-Kreis     15-34    NaN 2020-10-28
518     Baden-Württemberg  LK Alb-Donau-Kreis     15-34    NaN 2020-10-30
519     Baden-Württemberg  LK Alb-Donau-Kreis     15-34    NaN 2020-11-01
…                   …                  …       …     …        …
415484          Thüringen             SK Jena       NaN      F 2020-12-29
415485          Thüringen             SK Jena       NaN      M 2020-12-28
415486          Thüringen             SK Jena       NaN      M 2020-12-29
415487          Thüringen             SK Jena       NaN      M 2020-12-30
416313          Thüringen           SK Weimar       NaN      F 2020-12-31

        cases  deaths  recovered
200         1       0          1
201         1       0          1
517         1       0          1
518         2       0          2
519         1       0          1
…       …     …        …
415484      1       0          0
415485      1       0          0
415486      1       0          0
415487      2       0          0
416313      1       0          0

[11402 rows x 8 columns]
```

```
[73]: df_dmo[(df_dmo["gender"].isnull()) | (df_dmo["age_group"].isnull())]
```

```
[73]: Empty DataFrame
      Columns: [state, gender, age_group, population]
      Index: []
```

#This data set has no “NaN” values

```
[74]: #getting the sum of the 'NA' values

      df_covid[(df_covid["gender"].isnull()) | (df_covid["age_group"].isnull())].sum()
```

```
[74]: state        Baden-WürttembergBaden-WürttembergBaden-Württe…
      county       LK Alb-Donau-KreisLK Alb-Donau-KreisLK Alb-Don…
      cases                                                     17160
      deaths                                                       65
      recovered                                                 12189
      dtype: object
```

```
[75]: #filling age_group "NA" values with most frequent values

      group = df_covid.age_group.value_counts().idxmax()
```

```
df_covid.age_group.fillna(group, inplace = True)
```

```
[76]: #filling missing values of gender column (half with 'M' and other half with 'F')

gender = df_covid.gender.isna()
gen = df_covid.gender.loc[gender].sample(frac = 0.5).index  #generating random␣
 ↪rows
df_covid.loc[gen, 'gender'] = 'M'
df_covid.gender.fillna('F', inplace = True)
```

##Now the ‘NA’ values is filled up. We can Check it by calling the isnull() function

```
[77]: df_covid[(df_covid["gender"].isnull()) | (df_covid["age_group"].isnull())]
```

```
[77]: Empty DataFrame
Columns: [state, county, age_group, gender, date, cases, deaths, recovered]
Index: []
```

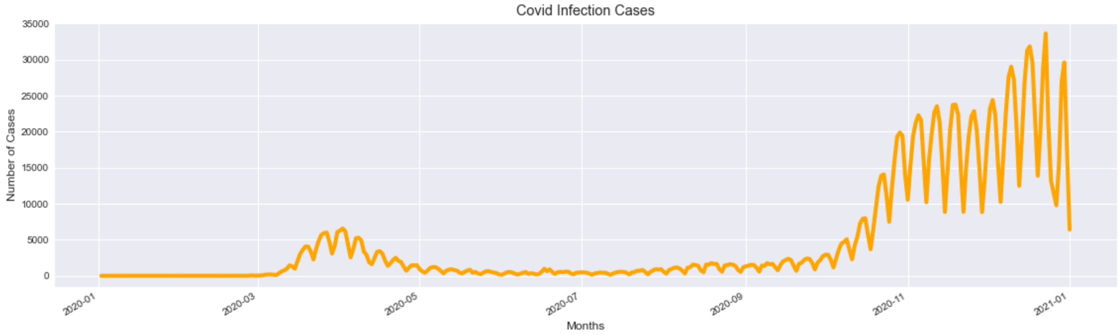
```
[78]: #getting mean,max, total values from dataset

df_covid.describe()
```

```
[78]:           cases    deaths  recovered
count 416314.00 416314.00  416314.00
mean       4.22      0.08       3.29
std        6.71      0.38       5.62
min        0.00      0.00      -2.00
25%        1.00      0.00       1.00
50%        2.00      0.00       1.00
75%        4.00      0.00       3.00
max      206.00     13.00     206.00
```

##We will start analyzing out data. For that purpose we will use differnt kind of charts, plots from Matplotlib, Seaborn and GeoPandas library. Let’s see what we can do with the mentioned library

```
[79]: #plotting the covid cases according to the month

daily_cases = df_covid.groupby("date").sum()


sns.set_style("darkgrid")
#sns.color_palette("cool")

plt.figure(figsize=(16,5))
plt.title("Covid Infection Cases")
sns.lineplot(data = daily_cases["cases"], color = "orange")
```

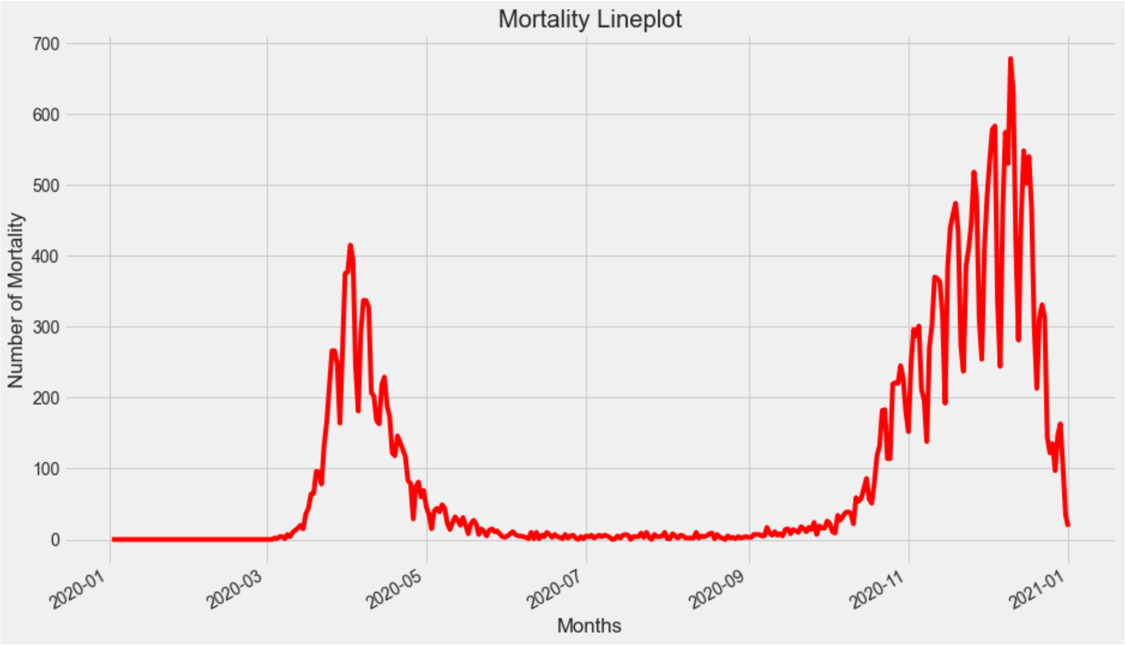
```
plt.ylabel("Number of Cases")
plt.xlabel("Months")

plt.style.use("fivethirtyeight")

plt.gcf().autofmt_xdate()
plt.tight_layout()
plt.show()
```



##From the lineplot of covid_cases we can say that, infection rate were almost 0 between January to March. Between March and May the number of infected people increases compare to the first 3 months. After that, till October the amount of cases were under control before significantly increasing during the period of November to January. This period is called the second wave of Covid.

[80]:

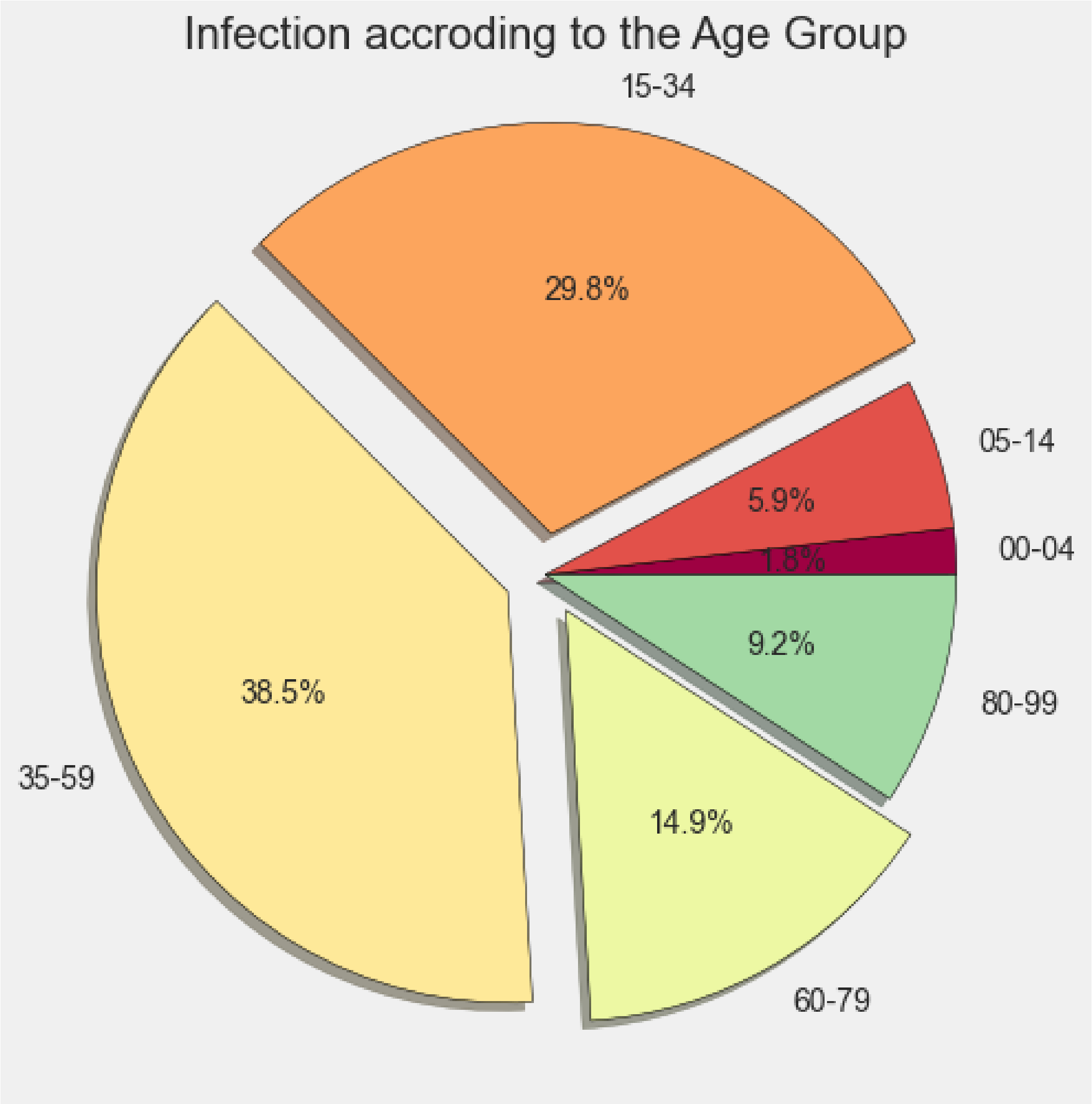
```
#daily mortality

plt.figure(figsize=(14,9))
plt.title("Mortality Lineplot")
sns.lineplot(data = daily_cases["deaths"], color = "red")
plt.ylabel("Number of Mortality")
plt.xlabel("Months")
plt.gcf().autofmt_xdate()
plt.show()
```



##The death cases followed the same graph as covid cases but the numbers were significantly low.The Death cases reached highest level of the graph during December and January.

[81]:
```
#which age groups are most affected by the virus

plt.style.use("fivethirtyeight")
plt.tight_layout()

by_age = df_covid.groupby("age_group").sum()
by_age.sort_values("cases",ascending = False)

cmap = plt.get_cmap("Spectral")
colors = [cmap(i) for i in np.linspace(0, 1, 8)]
explode = [0,0,0.1,0.1,0.1,0]

plt.figure(figsize = (10,8))
plt.title("Infection accroding to the Age Group")
cases_pie = plt.pie(by_age.cases, labels = by_age.index, autopct = "%1.
 ↪1f%%",shadow = True , colors = colors
                        ,wedgeprops={'edgecolor':'black'},explode = explode)
```

```
<Figure size 432x288 with 0 Axes>
```

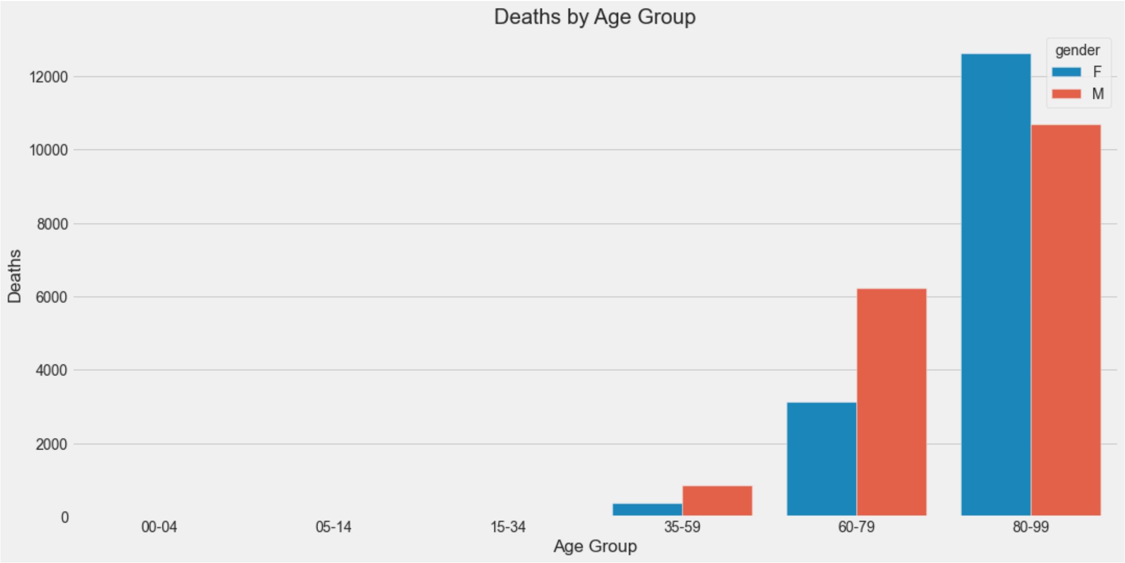


##From the pie chart we can tell that people of age group of 35-59 are the most infected age group. But covid was not nice to the elderly and younger age groups.

[82]:

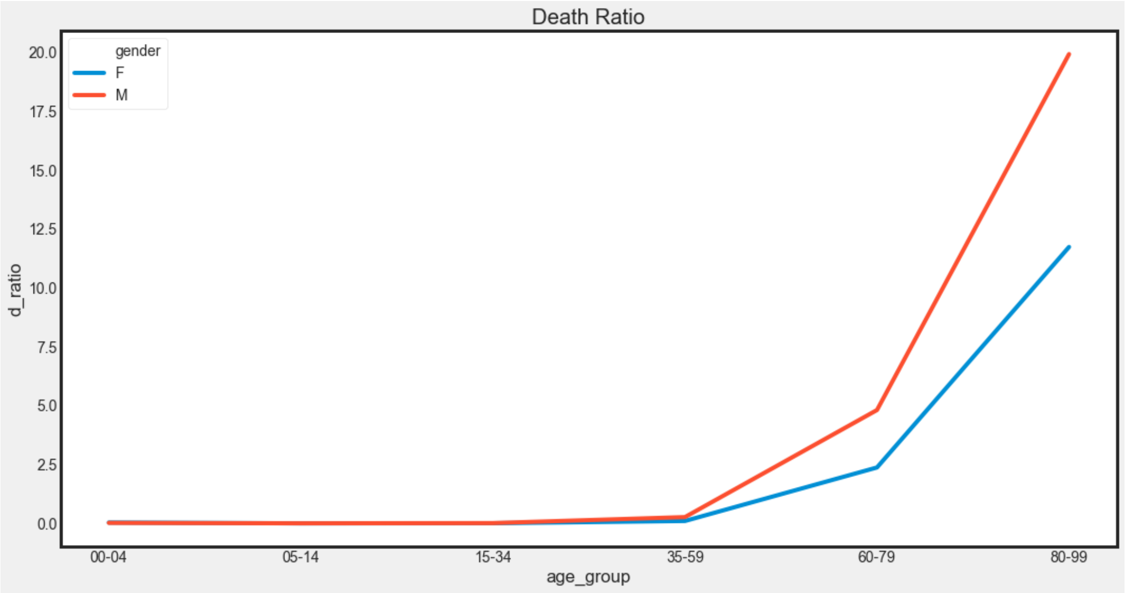
```
#How the number of deaths is distributed over different age groups of both␣
 ↪genders

covid = df_covid.groupby(['age_group','gender'], as_index = False).sum()

plt.figure(figsize=(16,8))
sns.barplot(y = covid.deaths, x = covid.age_group, hue = covid.gender, data =␣
 ↪covid)

plt.style.use("fivethirtyeight")
plt.title('Deaths by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Deaths')
```

[82]:
```
Text(0, 0.5, 'Deaths')
```



##This barplot indicates that females in the 80-99 age group died more by the virus than males. The ratio is half and totally opposite in the age group of 60-79.

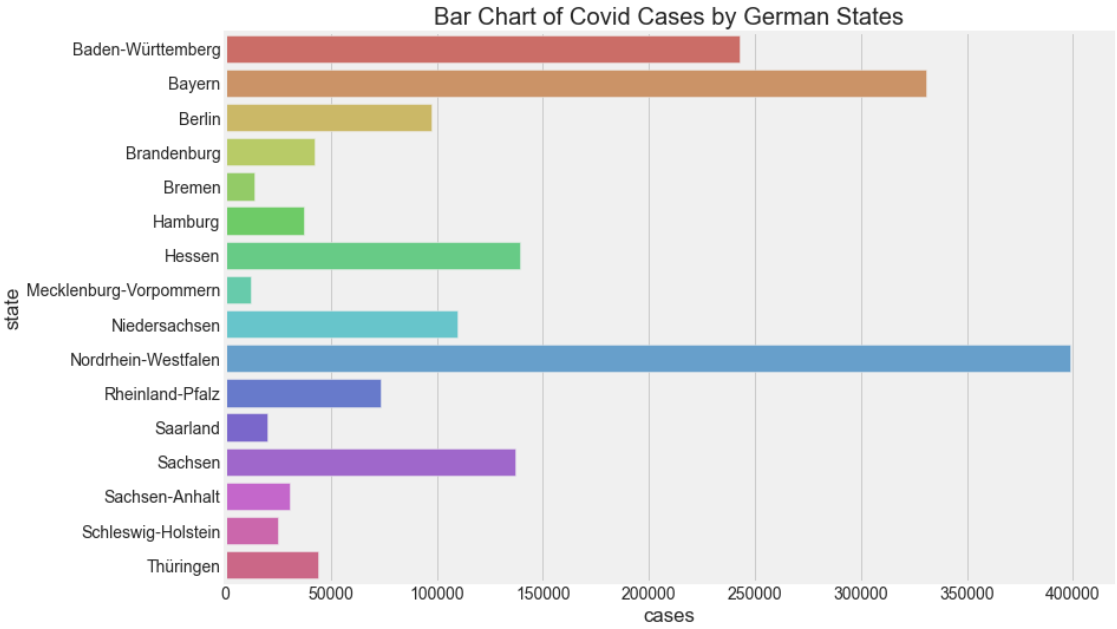
[83]:
```
#death ratio by gender

gender = df_covid.groupby(by=["age_group","gender"]).sum().reset_index()
gender["d_ratio"] = 100 * gender["deaths"] / gender["cases"]

plt.figure(figsize=(15,8))
sns.set_style("ticks")
sns.lineplot(data = gender,x="age_group",y="d_ratio",hue = "gender")
plt.title("Death Ratio")

plt.tight_layout()
plt.show()
```



#This lineplot illustrates that death ratio is higher in elderly male age-group compare to female age groups.

[84]:

```
#states comparison

by_state = df_covid.groupby("state").sum()
by_state.sort_values("cases",ascending = True)
#width = 0.25

plt.figure(figsize=(12,8))
plt.style.use("fivethirtyeight")
plt.tight_layout()

plt.title("Bar Chart of Covid Cases by German States")
#plt.xlabel("Number of Total Cases")
#plt.ylabel("Name of State")
sns.barplot(by_state.cases, by_state.index, palette= 'hls')
#plt.legend()
plt.show()
```



##We can see from the barplot above that, Nordrhein-Westfalen has got the most coronavirus cases. The least infected state is Hessen and Bremen. Bayern and Baden-Württemberg are also in the track of most infection

[85]:
```
df_mp.head()
```

[85]:
```
   ADE  RS          RS_0                  GEN  \
0    2  02  020000000000              Hamburg
1    2  03  030000000000        Niedersachsen
2    2  04  040000000000               Bremen
3    2  05  050000000000  Nordrhein-Westfalen
4    2  06  060000000000               Hessen

                                            geometry
0  MULTIPOLYGON (((3578695.661 5955304.456, 35781…
1  MULTIPOLYGON (((3354775.046 5942939.764, 33546…
2  MULTIPOLYGON (((3468658.496 5898364.974, 34702…
3  POLYGON ((3477450.781 5820982.368, 3479895.578…
4  POLYGON ((3535084.230 5721608.644, 3535279.888…
```

[86]:
```
#dropping unnecessary columns from dataset

df_mp = df_mp.drop(columns = ["ADE", "RS" ,"RS_0"])
```

[87]:
```
df_mp.head()
```

```
[87]:                    GEN                                           geometry
      0              Hamburg  MULTIPOLYGON (((3578695.661 5955304.456, 35781…
      1        Niedersachsen  MULTIPOLYGON (((3354775.046 5942939.764, 33546…
      2               Bremen  MULTIPOLYGON (((3468658.496 5898364.974, 34702…
      3  Nordrhein-Westfalen  POLYGON ((3477450.781 5820982.368, 3479895.578…
      4               Hessen  POLYGON ((3535084.230 5721608.644, 3535279.888…
```

```
[88]: state_covid = df_covid.groupby(by = 'state',as_index=False).sum()
      state_demo = df_dmo[['state','population']].groupby(by='state',as_index =␣
       ↪False).sum()
      df_state = df_mp.merge(state_covid, how = "left", left_on = "GEN", right_on =␣
       ↪"state")  #merging 2 dataset together and alligning the columns
      df_state = df_state.merge(state_demo, how = "left", left_on = "GEN", right_on =␣
       ↪"state")
      df_state = df_state.drop(columns = ["state_x", "state_y"]) #dropping state␣
       ↪column to avoid multiple same value column existence

      df_state["case_ratio"] = df_state["cases"] * (1000 / df_state ["population"]) ␣
       ↪#getting case ratio per 1000 in a new column
      df_state["d_ratio"] = df_state["deaths"] * (1000 / df_state ["population"])␣
       ↪#getting death case ratio per 1000 in a new column
      df_state["d_case_ratio"]= 100 * df_state["deaths"] / df_state["cases"] ␣
       ↪#getting death-case ratio in a new column

      df_state.set_index("GEN", inplace = True)
      #state_covid
      #state_demo
      df_state.head()
```

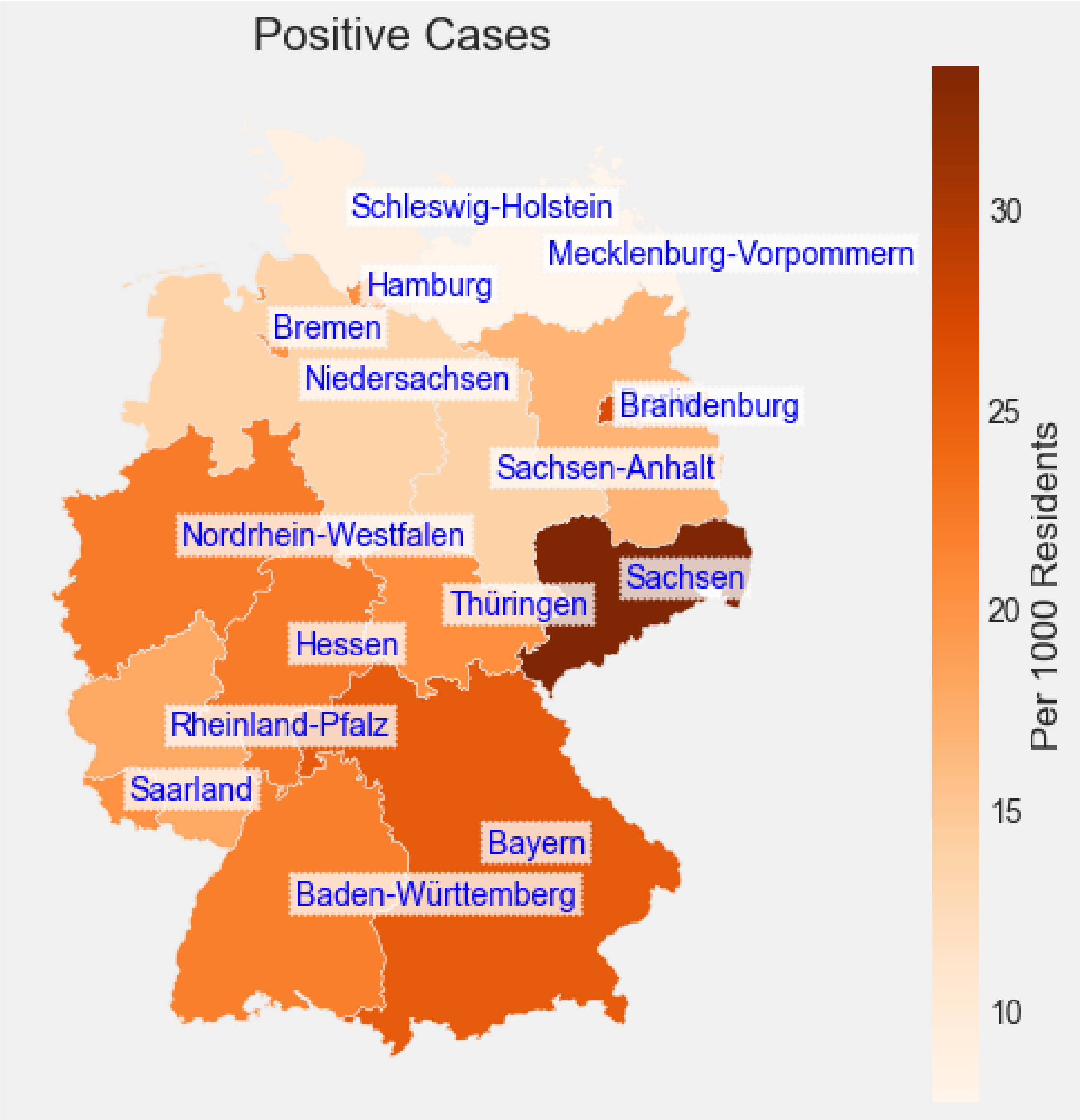
```
[88]:                                                             geometry  \
      GEN
      Hamburg              MULTIPOLYGON (((3578695.661 5955304.456, 35781…
      Niedersachsen        MULTIPOLYGON (((3354775.046 5942939.764, 33546…
      Bremen               MULTIPOLYGON (((3468658.496 5898364.974, 34702…
      Nordrhein-Westfalen  POLYGON ((3477450.781 5820982.368, 3479895.578…
      Hessen               POLYGON ((3535084.230 5721608.644, 3535279.888…

                            cases  deaths  recovered  population  case_ratio  \
      GEN
      Hamburg               37286     658      27236     1841179       20.25
      Niedersachsen        109797    2016      92551     7982448       13.75
      Bremen                13700     201      11380      682986       20.06
      Nordrhein-Westfalen  398661    6701     326307    17932651       22.23
      Hessen               139349    2917     109223     6265809       22.24

                           d_ratio  d_case_ratio
      GEN
```

```
Hamburg                 0.36          1.76
Niedersachsen           0.25          1.84
Bremen                  0.29          1.47
Nordrhein-Westfalen     0.37          1.68
Hessen                  0.47          2.09
```

#We have merged 3 different dataset in a single dataset. We have added some new columns in our new dataset. It will help us to visulize our geographical overview.case_ratio, death ratio and death case ratio column will be used in geographical graphs.

[89]:

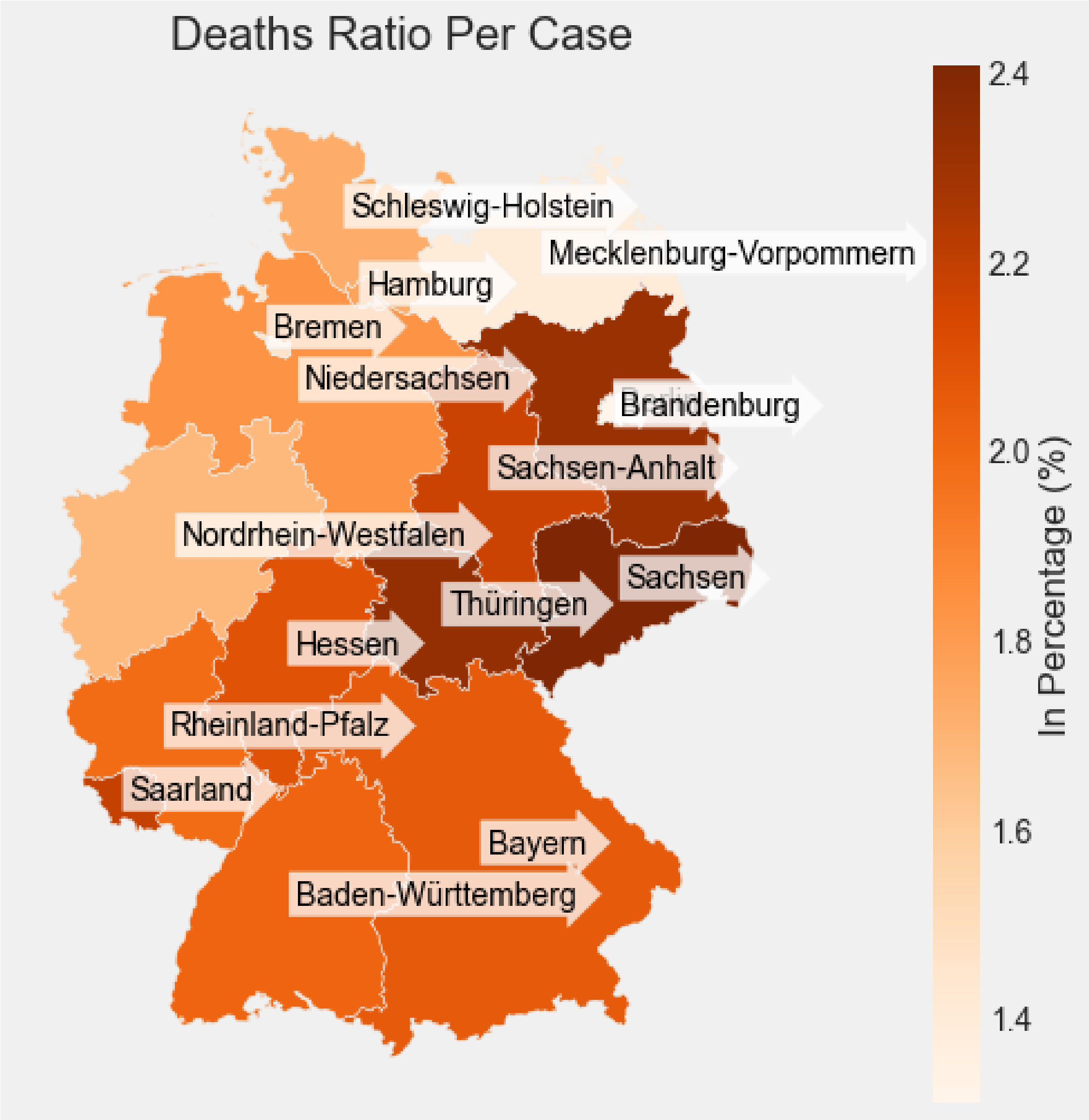
```
fig, (ax) = plt.subplots(1, figsize=(20,8), edgecolor = 'black')



df_state.plot(column = "case_ratio", legend = True, legend_kwds = {"label":␣
 ↪"Per 1000 Residents",
                                                                           ␣
 ↪"orientation" : "vertical"}, cmap = "Oranges", ax = ax)

for i, geo in df_state.centroid.iteritems():
    ax.annotate(s=i, xy=[geo.x , geo.y], color = "blue", bbox = dict(boxstyle =␣
 ↪'sawtooth, pad =0.2' , fc = 'white', alpha = 0.75))

    ax.set_title("Positive Cases")

ax.axes.get_xaxis().set_visible(False)
ax.axes.get_yaxis().set_visible(False)
```



```
[ ]:
```

```
[90]: fig, (ax1) = plt.subplots(1, figsize=(20,8), edgecolor = 'black')

      df_state.plot(column="d_case_ratio",legend = True, legend_kwds={"label": "In␣
       ↪Percentage (%)",
                                                                              ␣
       ↪"orientation": "vertical"}, cmap = "Oranges", ax = ax1)

      for i, geo in df_state.centroid.iteritems():
```

```
    ax1.annotate(s=i, xy=[geo.x, geo.y], color = "black", bbox = dict(boxstyle␣
 ↪= 'rarrow, pad =0.2', fc = 'white', alpha = 0.75))
ax1.set_title("Deaths Ratio Per Case")

ax1.axes.get_xaxis().set_visible(False)
ax1.axes.get_yaxis().set_visible(False)

plt.show()
```



#Using the geomatric value of our dataset we have tried to visualize all the states of Germany according to Positive cases and Death ratio per case. They have been calculated above. Because of using multiple dataset for displaying our geomatric data there might be a sligh change between state comparison bar graph and geo graph. The data displayed above might seem diverse.

1 THANK YOU

[ ]:

```
```